Argonne NATIONAL LABORATORY

DEVELOPING APPLICATIONS FOR AURORA

Argonne Leadership Computing Facility

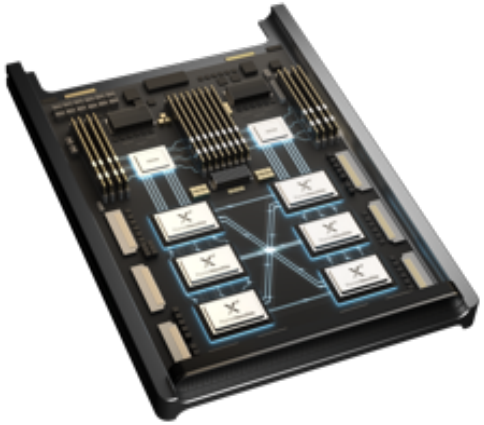
P3HPC FORUM, SEPT 1 - 2

SCOTT PARKER

www.anl.gov

Aurora: A High-level View

- Intel-Cray machine arriving at Argonne in 2021:
  - Sustained Performance > 1 Exaflops
  - Greater than 10 PB of total memory
- Node has Intel Xeon processors and Intel X^e GPUs:
  - 2 Xeons (Sapphire Rapids)
  - 6 GPUs (Ponte Vecchio [PVC])
  - Unified Memory Architecture across CPUs and GPUs
- Cray Slingshot fabric and Shasta platform:
  - 8 endpoints per node
- Novel high-performance filesystem:
  - Distributed Asynchronous Object Store (DAOS)
    - ≥ 230 PB of storage capacity
    - Bandwidth of > 25 TB/s
  - Lustre
    - 150 PB of storage capacity
    - Bandwidth of ~1TB/s



Three Pillars: Simulation, Data, and Learning

Simulation	Data	Learning
HPC Languages	Productivity Languages	Productivity Languages
Directives	Big Data Stack	DL Frameworks
Parallel Runtimes	Statistical Libraries	Statistical Libraries
Solver Libraries	Databases	Linear Algebra Libraries

Compilers, Performance Tools, Debuggers
Math Libraries, C++ Standard Library, libc
I/O, Messaging
Containers, Visualization
Scheduler
Linux Kernel, POSIX

Aurora Applications Overview

- Early applications for Aurora come from two programs:
  - The Argonne Early Science Program (ESP) : 20 projects
  - The DOE Exascale Computing Project (ECP) : 15 projects
- Some projects contain multiple codes
  - overall at least 50 codes are under development for Aurora
- Almost all projects involve teams from outside Argonne

Aurora ECP and ESP Projects and Applications

Project	Type	Applications
Candle	ESP, ECP	Candle/Uno
Many Body	ESP	BerkelyGW, FHI-aims, Quantum Expresso
LHC-ATLAS	ESP	FastCaloSim, MadGraph
DL Fusion	ESP	FusionDL
ExaSky, Extreme Scale Cosmo	ECP, ESP	HACC, NYX,
Connectomics	ESP	Flood Fill Network, AlignTK, mb_aligner
LatticeQCD	ESP, ECP	MILC, QUDA, Grid, Chroma
NAMD	ESP	NAMD
QMCPACK	ESP, ECP	QMCPACK
ExaFEL	ECP	Psana, CCTBX, MTIP, nanoBragg
EQSim	ECP	SW4, ESSI
ExaSMR	ECP	NekRS, OpenMC

Aurora ECP and ESP Projects and Applications

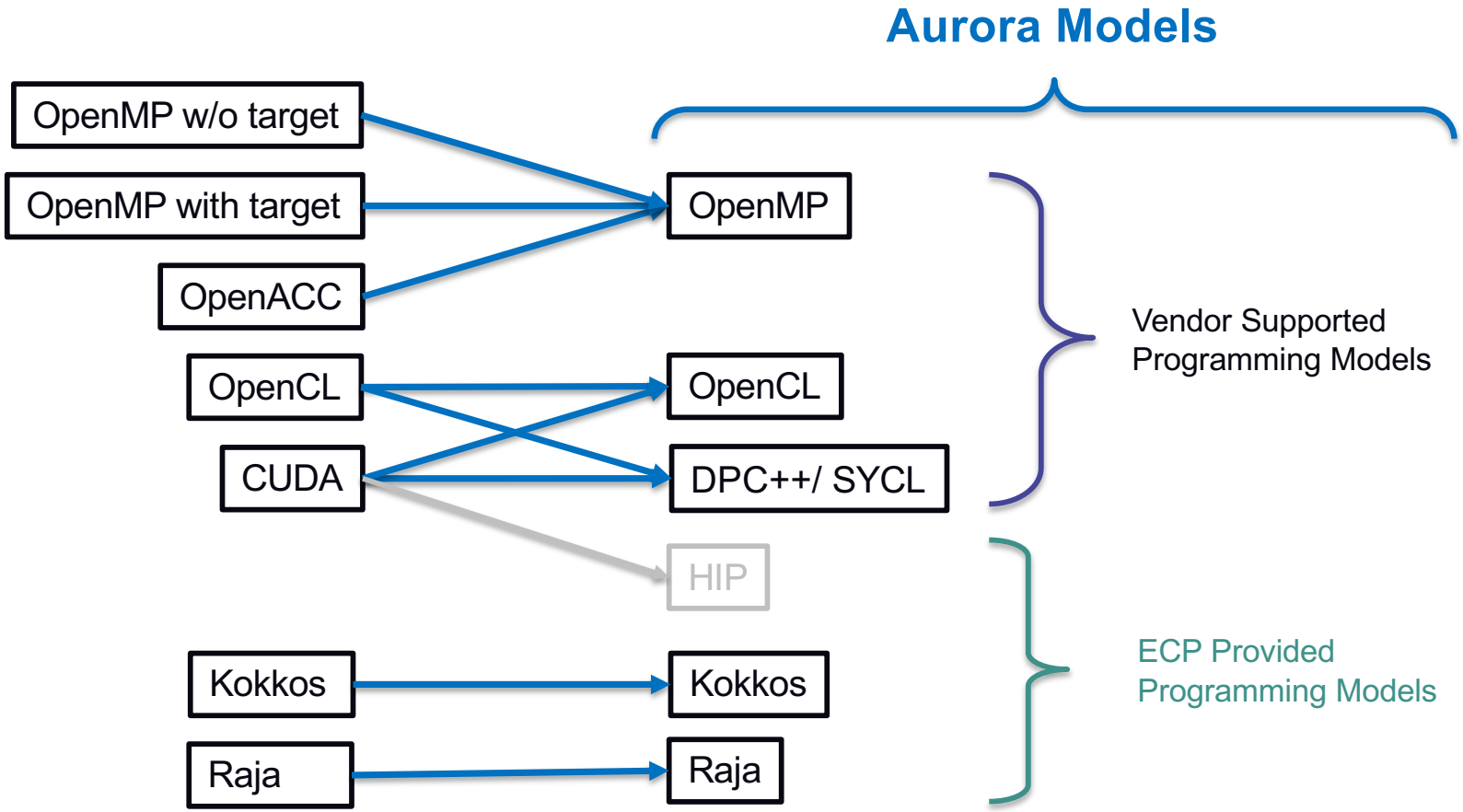
Project	Type	Applications
GAMESS	ECP	GAMESS
NWCHEMEX, Catalysis	ECP, ESP	NWChemEx
ExaStar	ECP	Castro, Flash
E3SM	ESP	ES3M-MMF
NQQMC_RMD	ESP	QXMD, RMD
PHASTA	ESP	PHASTA
Multiphysics	ESP	HARVEY
MFXI-EXA	ECP	MFIX-Exa
EXAALT	ECP	LAMMPS, LATTE
WDMApp, XGC ESP	ECP, ESP	XGC, GENE, GEM
Uintah ESP	ESP	Uintah
ExaWind	ECP	Nalu-Wind, AMR-Wind

Languages Used By Aurora Applications

- Aurora will the provide Fortran, C, C++, and Python language
- The set of applications being prepared for Aurora are using all of these languages
- Some applications use more than one language
- C++ is the most used language, followed by Fortran
- Usage of Python is growing particularly for data and learning applications

Language	Application Count
C++	35
Fortran	23
C	12
Python	12

Choosing an Aurora Programming Model



Programming Model Adoption By Aurora Applications

- All of the Aurora programming models are being used by applications
- Some applications are using or exploring more than one programming model
- OpenMP the most commonly used model, followed by DPC++/SYLC and Kokkos

Programming Model	Application Count
OpenMP	19
DPC++/SYCL	18
Kokkos	6
RAJA	1
OpenCL	1
OCCA	1

OpenMP 5

- OpenMP 5 constructs will provide directives based programming model for Intel GPUs
- Available for C, C++, and Fortran
- A portable model expected to be supported on a variety of platforms (Aurora, Frontier, Perlmutter, …)
- Optimized for Aurora
- For Aurora, OpenACC codes could be converted into OpenMP

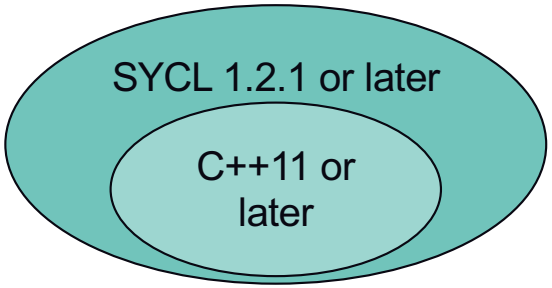
https://www.openmp.org/

Multiple compilers will support a common set of OpenMP directives on GPUs

	LLVM/Clang 10	AMD (mostly tracks LLVM)	Cray (CCE 10)	IBM (XL V16.1.6)	Intel (Approximately 2021 timeframe)	NVIDIA/PGI (Early 2021 for a production release)
Levels of parallelism	2 (teams, parallel) (11: 3 (teams, parallel, simd))	2 (teams, parallel)	2 (teams, parallel or simd)	2 (teams, parallel)	3 (teams, parallel, simd)	2 (teams, parallel)
OpenMP directive						
target	✓	✓	✓	✓	✓	✓
declare target	✓	✓	✓	✓	✓	✓
map	✓	✓	✓	✓	✓	✓
target data	✓	✓	✓	✓	✓	✓
target enter/exit data	✓	✓	✓	✓	✓	✓
target update	✓	✓	✓	✓	✓	✓
teams	✓	✓	✓	✓	✓	✓
distribute	✓	✓	✓	✓	✓	✓
parallel	✓	✓	✓ (may be inactive)	✓	✓	✓
for/do	✓	✓	✓	✓	✓	✓
reduction	✓	✓	✓	✓	✓	✓
simd	simdlen(1) (11: honored with hint)	✓ (on host)	✓	✓ (accepted and ignored)	✓	✓ simdlen(1)
atomic	✓	✓	✓	✓	✓	✓
critical	✓	✓	✓	✓	✓	✘
sections	✓	✓	✓	✓	✓	✘
master	✓	✓	✓	✓	✓	✓
single	✓	✓	✓	✓	✓	✓
barrier	✓	✓	✓	✓	✓	✓
declare variant	✓	✓	(support planned for CCE 11)	✘	✓	✓

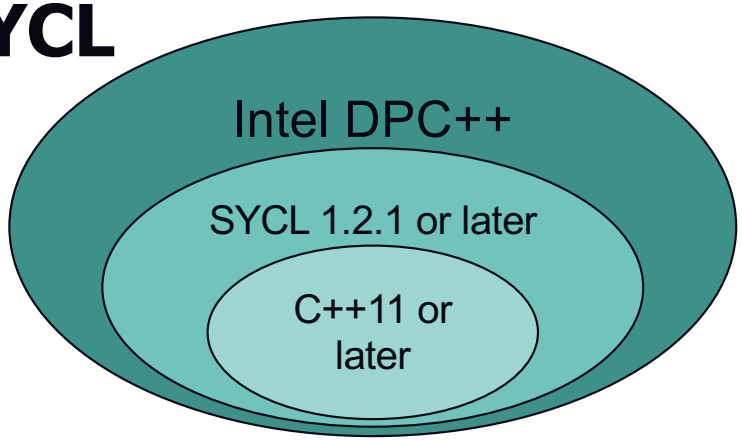
DPC++ (Data Parallel C++) and SYCL

- SYCL
  - Khronos standard specification
  - SYCL is a C++ based abstraction layer (standard C++11)
  - Builds on OpenCL **concepts** (but single-source)
  - *SYCL is designed to be as close to standard C++ as possible*
- Current Implementations of SYCL:
  - ComputeCPP™ (www.codeplay.com)
  - Intel SYCL (github.com/intel/llvm)
  - triSYCL (github.com/triSYCL/triSYCL)
  - hipSYCL (github.com/illuhad/hipSYCL)
  - **Runs on today's CPUs and nVidia, AMD, Intel GPUs**



DPC++ (Data Parallel C++) and SYCL

- SYCL
  - Khronos standard specification
  - SYCL is a C++ based abstraction layer (standard C++11)
  - Builds on OpenCL **concepts** (but single-source)
  - *SYCL is designed to be as close to standard C++ as possible*
- Current Implementations of SYCL:
  - ComputeCPP™ (www.codeplay.com)
  - Intel SYCL (github.com/intel/llvm)
  - triSYCL (github.com/triSYCL/triSYCL)
  - hipSYCL (github.com/illuhad/hipSYCL)
  - **Runs on today's CPUs and nVidia, AMD, Intel GPUs**
- DPC++
  - Part of Intel oneAPI specification
  - Intel extension of SYCL to support new innovative features
  - Incorporates SYCL 1.2.1 specification and Unified Shared Memory
  - Adds language or runtime extensions as needed to meet user needs



Extensions	Description
Unified Shared Memory (USM)	defines pointer-based memory accesses and management interfaces.
In-order queues	defines simple in-order semantics for queues, to simplify common coding patterns.
Reduction	provides reduction abstraction to the ND-range form of parallel_for.
Optional lambda name	removes requirement to manually name lambdas that define kernels.
Subgroups	defines a grouping of work-items within a work-group.
Data flow pipes	enables efficient First-In, First-Out (FIFO) communication (FPGA-only)

https://spec.oneapi.com/oneAPI/Elements/dpcpp/dpcpp_root.html#extensions-table

OpenCL

- Open standard for heterogeneous device programming (CPU, GPU, FPGA)
  - Utilized for GPU programming
- Standardized by multi-vendor Khronos Group, V 1.0 released in 2009
  - AMD, Intel, nVidia, …
  - Many implementations from different vendors
- Intel implementation for GPU is Open Source (https://github.com/intel/compute-runtime)
- SIMT programming model
  - Distributes work by abstracting loops and assigning work to threads
  - Not using pragmas / directives for specifying parallelism
  - Similar model to CUDA
- Consists of a C compliant library with kernel language
  - Kernel language extends C
  - Has extensions that may be vendor specific
- Programming aspects:
  - Requires host and device code be in different files
  - Typically uses JIT compilation
- Example: https://github.com/alcf-perfengr/alcl

RAJA

- RAJA is a collection of C++ software abstractions that enable architecture portability for HPC applications.
- The goals of RAJA are to:
  - Make existing applications portable with minimal disruption
  - Provide a model for new applications so that they are portable from inception
- RAJA targets portable, parallel loop execution by providing building blocks that extend the generally accepted parallel for idiom
- RAJA is being implemented for Aurora by ECP and Argonne
  - The SW4 application and much of the RAJA Perf Suite are working
  - Initial performance looks promising
  - Work remains to support all RAJA features and resolve performance issues

Kokkos

- Kokkos is a C++ programming model for developing performance portable applications
- Provides abstractions for both the parallel execution of code and for data management
- Kokkos on Aurora will be implemented using both OpenMP and DPC++ backends
  - OpenMP backend –
    - Passing Kokkos unit tests
    - Some application partially or fully working
  - DPC++ backend
    - Implementation underway
    - Currently waiting on resolution of some compiler issues

HIP For Aurora

- Exascale Computing Project (ECP) recently funded an effort to begin implementing HIP for Aurora
- Being done in collaboration with AMD
- Effort will leverage the fact that both HIP and Intel GPU compilers utilize the LLVM framework
- HIP will be implemented on top of Intel's Level Zero runtime layer
- Work is in its initial phases and the extent to which it will be a viable options for applications in the near term had not yet been determined

Preparing CUDA Codes For Aurora

- Portability is an issue for codes utilizing CUDA
  - CUDA is a proprietary model for programming nVidia GPUs
  - HIP has been developed by AMD to allow easy porting of CUDA codes to AMD GPUs
    - HIP code may also be run on nVidia GPUs
  - Initial work has started on a HIP implementation for Intel GPUs
- At present the long-term portability of CUDA is unclear

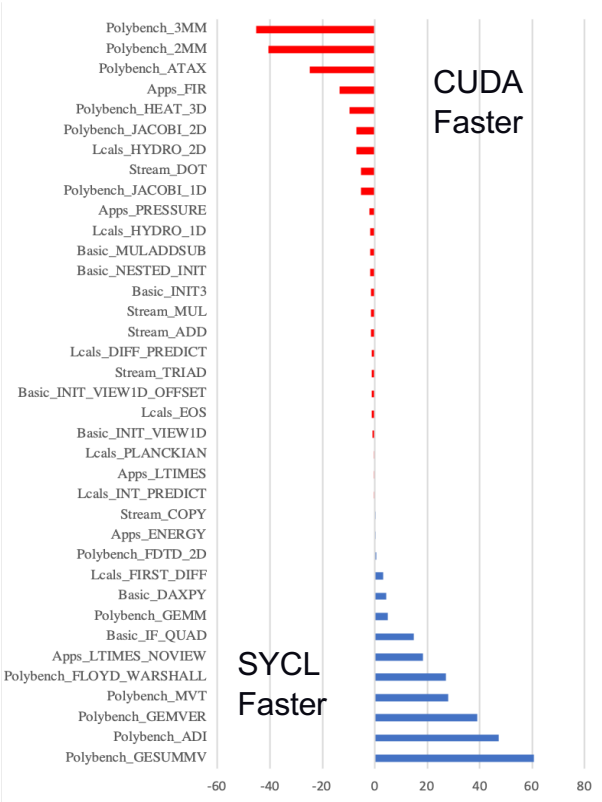
- Intel DPC++ Compatibility tool
  - Helps with the migration of CUDA code to DPC++
  - Currently in beta
  - Tool may not fully convert CUDA code to DPC++ and may require additional developer effort

Programming Model Performance

- A common concern is the relative performance of programming models on a given architecture and their efficiency across architectures
- Implementations of most programming models are still too immature to assess their long-term performance potential
- Early results suggest the Aurora programming models are capable of delivering roughly similar performance with some effort
- This suggest that programming model choice should not be made primarily on performance concerns

A Comparison of CUDA and SYCL using the RAJA Performance Suite

- The RAJA performance suite is a collection of performance benchmarks with RAJA and non-RAJA variants
- CUDA variants were ported to SYCL and performance compared on a V100 GPU using hipSYCL
- Figure shows the speedup of SYCL variant relative to the CUDA variant
- SYCL is showing competitive performance to CUDA on NVIDIA devices



From “Evaluating the Performance of the hipSYCL Toolchain for HPC Kernels on NVIDIA V100 GPUS” at SYCLCON 2020 by Brian Hommerding

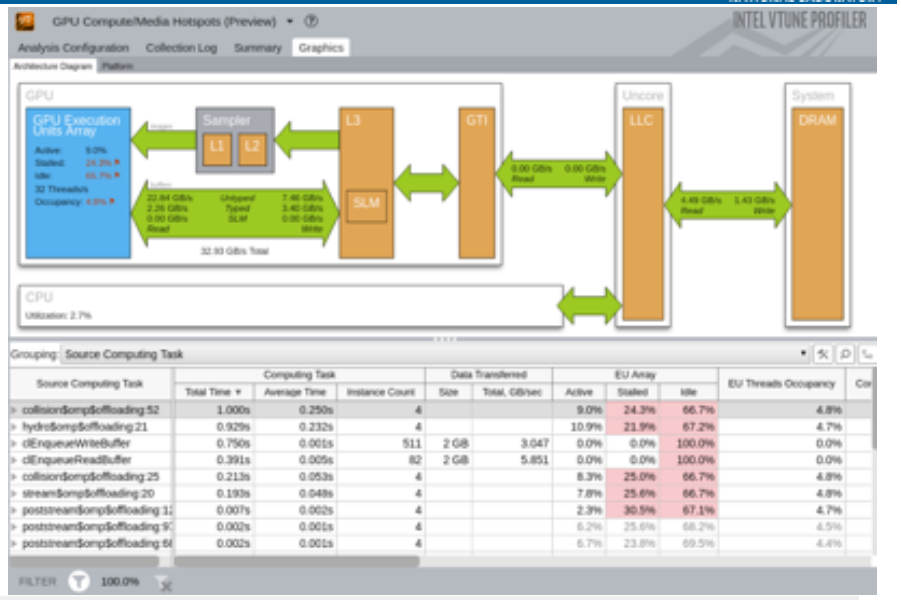
Intel Libraries

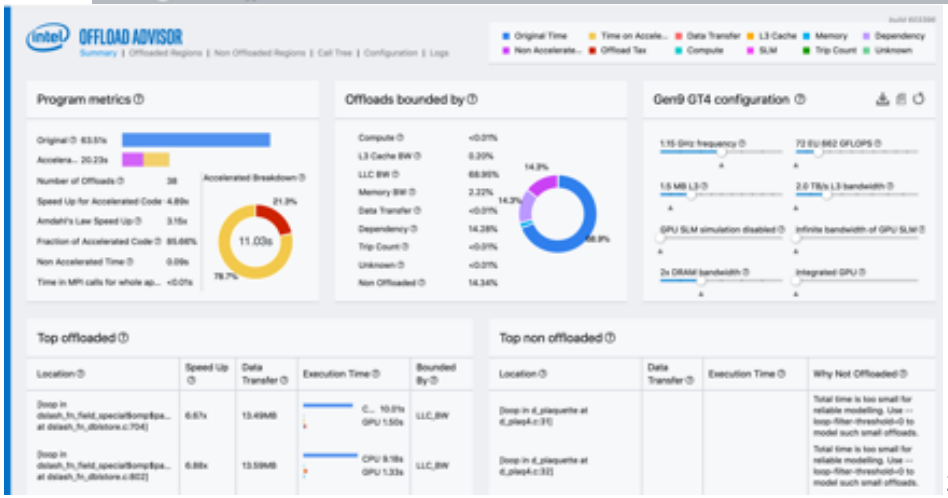
- The use of libraries and frameworks can simplify performance portability

- OneAPI MKL (oneMKL)
  - Highly tuned algorithms optimized for every Intel platform
  - FFT
  - Linear algebra (BLAS, LAPACK)
    - Sparse solvers
  - Statistical functions
  - Vector math
  - Random number generators
- OneAPI Deep Neural Network Library (oneDNN)
  - High Performance Primitives to accelerate deep learning frameworks
  - Powers Tensorflow, PyTorch, MXNet, Intel Caffe, and more
  - Running on Intel Gen9 GPUs today
- oneAPI Data Analytics Library (oneDAL)
  - Classical Machine Learning Algorithms
  - Easy to use one line daal4py Python interfaces
  - Powers Scikit-Learn

Intel VTune and Advisor

- Vtune Profiler
  - Widely used performance analysis tool
  - Currently supports analysis on Intel integrated GPUs
  - Will support future Intel GPUs
- Advisor
  - Provides roofline analysis
  - Offload analysis will identify components for profitable offload
    - Measure performance and behavior of original code
    - Model specific accelerator performance to determine offload opportunities
    - Considers overhead from data transfer and kernel launch





Aurora Testbeds

- Intel DevCloud
  - Provides free access to GPU hardware and oneAPI software
  - https://devcloud.intel.com/oneapi/get-started/
- Local Setup
  - Download Intel oneAPI public beta
    - https://software.intel.com/content/www/us/en/develop/tools/oneapi.html
  - Run on Intel CPU with integrated graphics
- Argonne JLSE testbeds for Aurora
  - 20 Nodes of Intel Xeons with Gen9 Iris Pro integrated GPU
  - DG1 nodes
  - Intel's Aurora oneAPI SDK [NDA required]

Argonne Joint Laboratory for System Evaluation


Questions?
Argonne
NATIONAL LABORATORY